MethodSCRIPT Examples - C#

PalmSens
Compact Electrochemical Interfaces

Last revision: March 22 2019

www.palmsens.com

1.1 Contents:

The examples in the */MethodSCRIPTExamples_C#* folder demonstrate basic communication with the EmStat Pico from a windows PC using C#. The examples show how to connect to the device, send MethodSCRIPTs to the device, run measurements on the device, read and parse the measurement data packages from the device and using simple plot objects to plot the data.

1.2 Examples:

1.2.1 Example 1: Basic Console Example (MSConsoleExample)

This example demonstrates how to implement USB serial communication to

- Establish a connection with the device
- Write a MethodSCRIPT to the device
- Read and parse measurement data packages from the device

This does not include error handling, method validation etc.

1.2.2 Example 2: Plot Example (MSPlotExample)

In addition to the basic communications as in the above example, this plot example demonstrates how to implement the plot object (using the OxyPlot library for windows forms).

1.2.3 Example 3: EIS Console Example (MSEISExample)

This console example demonstrates sending, receiving and parsing data for a simple EIS measurement.

1.2.4 Example 4: EIS Plot Example (MSEISPlotExample)

This example demonstrates the implementation of OxyPlot to show the EIS measurement response on Nyquist and Bode plots.

1.3 Communications

The example projects are built using Visual Studio 17.

1.3.1 Connecting to the device

The examples use the System.IO.Ports library for serial communication with the device. The read time out for the port is set to 7000ms. In case of measurements with long response times, the read time out can be set higher.

To prevent your program from crashing it is recommended to use a try catch sequence when connecting to a port, this way the port will be closed again when an exception occurs.

1.3.2 Sending a MethodSCRIPT

The methodSCRIPT can be read from a txt file stored in the *ScriptFilePath* and then sent to the device. In this example the MethodSCRIPT files are stored in the *scripts* directory in the same location as the application executable.

1.3.3 Receiving the measurement packages

Once the script file is sent to the device, the measurement packages can be read continuously from the device.

```
readChar = SerialPortEsP.ReadChar();
```

1.3.4 Parsing the measurement packages

The measurement data packages returned by the method *ReadResponseLine()*, can be parsed further to obtain the actual data values. Here's a set of data packages received from a Linear Sweep Voltammetry (LSV) measurement on a dummy cell with 10 kOhm resistance.

eM0000\n
Pda7F85F3Fu;ba48D503Dp,10,288\n
Pda7F9234Bu;ba4E2C324p,10,288\n
Pda806EC24u;baAE16C6Dp,10,288\n
Pda807B031u;baB360495p,10,288\n
*\n
\n

While parsing a measurement package, various identifiers are used to identify the type of package. For example, In the above sample,

1. 'e' is the confirmation of the "execute MethodSCRIPT" command.
2. 'M' marks the beginning of a measurement loop.
3. 'P' marks the beginning of a measurement data package.
4. "*\n" marks the end of a measurement loop.
5. "\n" marks the end of the MethodSCRIPT.

The data values to be received from a measurement can be sent through 'pck' commands in the MethodSCRIPT. Most techniques return the data values Potential (set cell potential in V) and Current (measured current in A). These can be sent with the MethodSCRIPT.

In case of Electrochemical Impedance Spectroscopy (EIS) measurements, the following *variable types* can be sent with the MethodSCRIPT and received as measurement data values.

- Frequency (set frequency in Hz)
- Real part of complex Impedance (measured impedance Ohm)
- Imaginary part of complex Impedance (measured impedance in Ohm)

The following metadata values if present can also be obtained from the data packages.

- CurrentStatus (OK, underload, overload, overload warning)
- CurrentRange (the current range in use at the moment)
- Noise (Noise)

1.3.4.1 Parsing the measurement data packages

Each measurement data package begins with the header 'P' and is terminated by a '\n'. The measurement data package can be split into data value packages based on the delimiter ';'. Each of these data value packages can then be parsed separately to get the actual data values.

The type of data in a data package is identified by its variable type:

- The potential readings are identified by the string *da*
- The current readings are identified by the string *ba*
- The frequency readings are identified by the string *dc*
- The real impedance readings are identified by the string *cc*
- The imaginary impedance readings are identified by the string *cd*

For example, in the sample package seen above, the *variable types* are
da7F85F3Fu - da Potential reading and
ba48D503Dp,10,288 – ba current reading.

The following 8 characters hold the data value. The data value for the current reading (8 characters) from the above sample package is *48D503Dp*.

The SI unit prefix from the package can be obtained from the parameter value at position 8

In the above sample package, the unit prefix for current data is 'p' which is 1e-12 A.

After obtaining variable type and the data values from the package, the metadata values can be parsed if present.

1.3.4.2 Parsing the metadata values

The metadata values are separated based on the delimiter ',' and each of the values is further parsed to get the actual value.

The first character of each metadata value metaData[0] identifies the type of metadata.

'1' – status
'2' – Current range index
'4' - Noise

The status is 1 character hex bit mask. It is converted to int. The status can be obtained as shown in the code snippet below.

For example, in the above sample, the available metadata values for current data are,
10,288. The first metadata value is 10.

1 – metadata status – 0 indicates OK.

The metadata type current range is 2 characters long hex value. If the first bit high (0x80), it indicates a high speed mode current range.

The code below can be used to get current range bits from the package.

The hex value is then converted to int to get the current range string as shown below.

For example, in the above sample, the second metadata available is 288.

2 – indicates the type – current range
88 – indicates the hex value for current range index – 1mA. The first bit 8 implies that it is high speed mode current range.

Sample outputs:

The following are some sample data packages and their corresponding outputs.

- A measurement data package from LSV measurement on a dummy cell with 10 kOhm resistance.

  Pda7F85F3Fu;ba4BA99F0p,10,288

  Output: E (V) = -4.999E-01
  i (A) = -4.999E-01
  Status : OK
  CR : 1mA (High speed)

- A measurement data package from EIS (Electrochemical Impedance Spectroscopy) measurement on a dummy cell with Randles circuit (560 Ohm, 10 kOhm, 33 nF)

  Pdc8030D40;cc8088C7Cm,12,289;cdA9279D1u,12,289

  Output:
  Frequency (Hz) : 200000.00
  Z' (Ohm) : 2.000E+05 Status : Overload CR : 5mA
  Z'' (Ohm) : 2.000E+05 Status : Overload CR : 5mA

1.3.5 Plotting the response

The OxyPlot library from NuGet packages is used in this example for showing a simple plot of the measurement response parameters.

```
using OxyPlot;
using OxyPlot.Axes;
using OxyPlot.Series;
```

Here's a sample plot with the response from EmStat Pico for a Linear Sweep Voltammetry measurement on a dummy cell with 10 kOhm resistance.

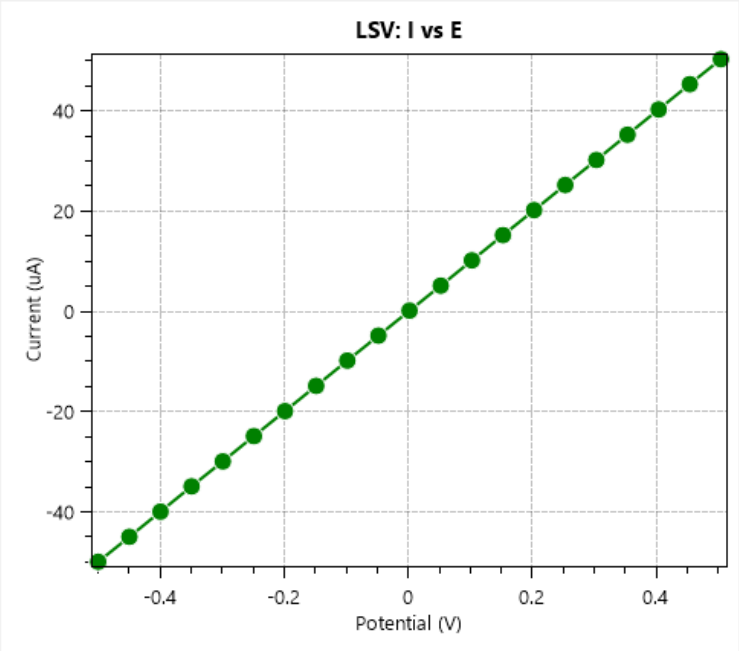

Fig: Sample plot LSV: I vs E

In case of an EIS (Electrochemical Impedance Spectroscopy) measurement, two separate plot objects and plot models are created. One for Nyquist plot and the other for Bode plot.

The plot type is set to *Cartesian* type in case of Nyquist plot model and *XY* type in case of Bode plot model. Nyquist plot axes are set to linear axes and Bode plot axes are set to logarithmic axes.

Here are the Nyquist and Bode plots with the response for an EIS measurement on a dummy cell with Randles circuit (560 Ohm, 10 kOhm, 33 nF).

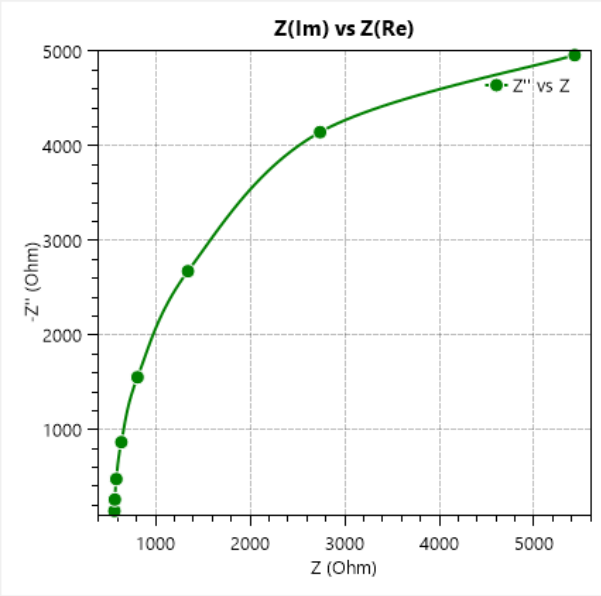

Fig: Sample plot EIS: Nyquist Plot

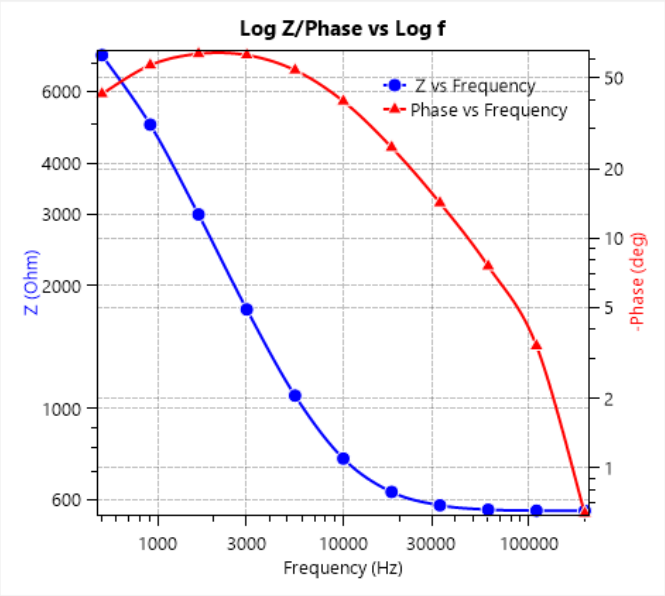

Fig: Sample plot EIS: Bode Plot